Criteria of Using VFIO Mdev (vs. Userspace DMA)

Kevin Tian, Ashok Raj
Purpose of Discussion

• VFIO mdev is the subdevice passthrough framework since 2017
  • Standard uAPI but require some emulation in kernel

• In-kernel emulation raises concern recently
  • When pushing a new mdev implementaion for IDXD device

• To be discussed in this session
  • Is it acceptable to put device emulation in kernel?
  • If no, what is the right approach?
  • If yes, how to prevent abuse of this framework?
VFIO MDEV

- In-kernel subdevice passthrough framework
  - Work queue, queue pair, context, etc.
  - vGPUs, Channel I/O devices, crypto devices, etc.

- Same uAPIs for device/subdevice passthrough
  - Existing VMMs just work!
  - Require some degree of emulation in host driver
    - Wrap subdevice as a virtual device (e.g. PCI)
    - Mediate control operations on subdevice

- Opens raised when using mdev in IDXD driver
  - Is it acceptable to put emulation in kernel?
    - E.g. low-risk pci cfg + simple mmio emulation...
  - If no, should vfio-mdev be deprecated and replaced by userspace DMA frameworks?
  - If yes, how to prevent abusing it as easy path to hook into virtualization stack?
• Allow userspace to directly ‘access’ device
  • E.g. mmap the command portal and submit workload

• Could be expanded for subdevice passthrough
  • Then contain vdev emulation in userspace
  • From ‘allow-access’ to ‘allow-control’
  • Meet virtualization demand
    • DMA map vs. vSVA, posted intr, live migration, etc.

• However,
  • Every driver requires specific uAPI support in all VMMs!
    • Although the real user is inside guest
  • Handling ‘control’ may increase uAPI complexity a lot
    • Modern ‘access’ uAPI is very simple (e.g. uacce)
    • ‘control’ uAPI might become a device API, to cover requirements from different guest Oses
  • Some degree of uAPI duplication

• It is not a net win over vfio-mdev!
  • VFIO wins on vendor agnostic uAPI
Proposal

• VFIO mdev has its merit as a standard subdevice passthrough framework
  • It’s fine if some driver wants to do its own way
  • But if vfio-mdev is used, we need criteria/process to catch any abuse

• Thoughts on preventing abuse:
  • A voting process similar to virtio
    • How to catch new attempt of mdev implementation?
  • A new mailing list for focused review/discussion of mdev implementations
    • Or, using KVM mailing list is sufficient?
  • Reduce code duplication
    • E.g. PCI Cfg space emulation, ioctl helpers, etc...
  • Explore moving some emulation to userspace
  • ...
Backup
History of VFIO MDEV

• Initial discussion ([link](#)) about a common mdev framework for vGPUs
• Converged proposal presented in KVM forum 2016 ([link](#))
• Vfio-mdev debuted in kernel 4.10 (2017), with KVMGT as the 1st user
• Other mdev implementations came in following years (s390 channel I/O devices, crypto devices, etc.)
• Recent hardware assistance (e.g. Intel Scalable IOV) allows reduced complexity and increased scalability ([link](#))